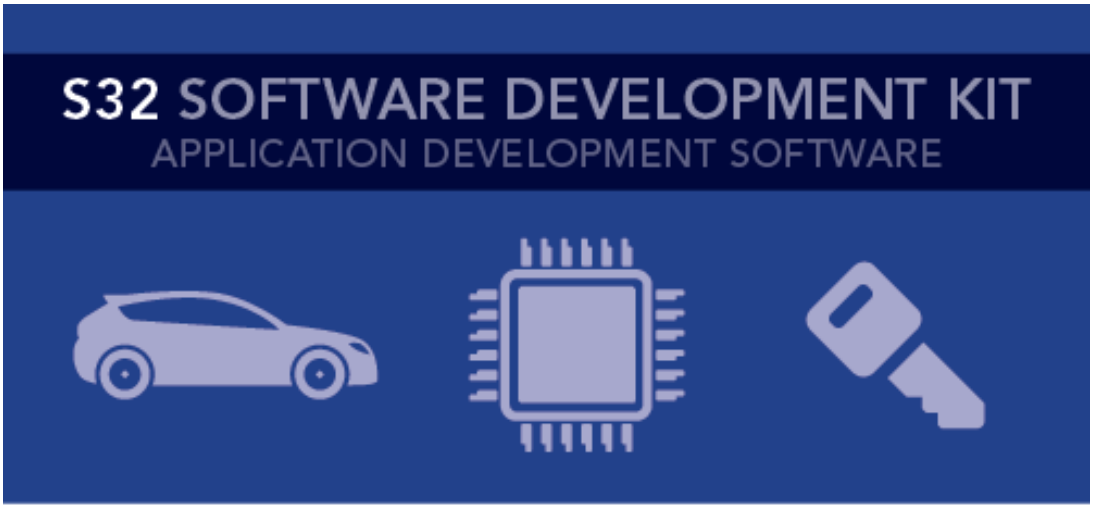

S32 SDK for S32K116 Release Notes

Version 1.8.7 EAR



Contents

1. Description

The S32 Software Development Kit (S32 SDK) is an extensive suite of peripheral abstraction layers, peripheral drivers, RTOS, stacks and middleware designed to simplify and accelerate application development on NXP S32K116 microcontrollers.

All software included in this release have EAR quality level in terms of features, testing and quality documentation, according to NXP software release criteria.

This SDK can be used standalone or it can be used with S32 Design Studio IDE (see Documentation).

Refer to *License(License.txt)* for licensing information and *Software content register(SW-Content-Register-S32-SDK.txt)* for the Software contents of this product. The files can be found in the root of the installation directory.

For support and issue reporting use the following ways of contact:

- Email to support@nxp.com
- NXP Community https://community.nxp.com/community/s32/s32k

Touch sense and ISELED libraries are available. Please contact with Sales representative or FAE for more information how to get them.

2. Software Contents

3.1 Drivers

- ADC
- CMP
- CRC
- CSEc
- DMA
- EIM
- ERM
- FLASH
- FLEXCAN
- FLEXIO (I2C, SPI, I2S, UART profiles)
- FTM
- LIN
- LPI2C
- LPIT
- LPSPI
- LPTMR
- LPUART
- MCU (ClockManager, InterruptManager, PowerManager)
- MPU
- PINS
- PDB
- RTC
- TRGMUX
- WDOG

3.2 PAL

- ADC
- CAN
- I2C
- I2S
- IC
- MPU
- OC
- PWM
- SECURITY
- SPI
- TIMING
- UART
- WDG

3.3 Middleware

- LIN stack - provides support for LIN 2.1, LIN 2.2 and J2602 communication protocols
- SBC - provides support for UJA1169 System Basis Chips

3. Documentation

- Quick start guide available in *"doc"* folder
- User and integration manual available at *"doc\Start_here.html".*
- Driver user manuals available in *"doc"* folder.

4. Examples

Type	Name	Description
Driver examples	adc_hwtrigger	Uses PDB to trigger an ADC conversion with a configured delay and sends the result to host via LPUART.
	adc_swtrigger	Uses software trigger to periodically trigger an ADC conversion and sends the result to host via LPUART.
	adc_pal_example	The application uses ADC PAL to trigger multiple executions of two groups of ADC conversions: first group configured for SW triggering and second group for HW triggering. For each execution of a group of conversions, an average conversion value is computed in SW, and the average value is printed on UART.
	can_pal	Shows the usage of the CAN PAL module with Flexible Data Rate
	cmp_dac	Configures the analog comparator to compare the input from the potentiometer with the internal DAC (configured to output half of the reference voltage) and shows the result using the LEDs found on the board.
	crc_checksum	The CRC is configured to generate the cyclic redundancy check value using 16 and 32 bits wide result.
	csec_keyconfig	The example demonstrates how to prepare the MCU before using CSEc(Key configuration).
	csec_flash_part	The example demonstrates how to prepare the MCU before using CSEc(flash partitioning).
	eim_injection	The purpose of this demo is to provide the user check able correction of ECC. Module EIM enable user addition error to RAM (low). And enable user can use module ERM to read address that user already error to region RAM. User seen RED_LED off when ERM read right address which EIM injected error.
	erm_report	The purpose of this driver application is to show the user how to use the EWM from the S32K116 using the S32 SDK API. This Example only debug equal Flash This example use module EIM to addition error to RAM and use module ERM to read address and notify interrupt .
	flash_partitioning	Writes, verifies and erases data on Flash.
	flexio_i2c	Demonstrates FlexIO I2C emulation. Use one instance of FlexIO and one instance of LPI2C to transfer data on the same board.
	flexio_spi	Demonstrates FlexIO SPI emulation for both master and slave configurations. Use one instance of FlexIO to

		instantiate master and slave drivers to transfer data on the same board.
	flexio_i2s	Demonstrates FlexIO I2S emulation for both master and slave configurations. Use one instance of FlexIO to instantiate master and slave drivers to transfer data on the same board.
	flexio_uart	Demonstrates FlexIO UART emulation for both TX and RX configurations. Use one instance of FlexIO to instantiate UART transmitter and receiver drivers to transfer data from/to the host.
	ftm_pwm	Uses FTM PWM functionality using a single channel to light a LED on the board. The light's intensity is increased and decreased periodically.
	ftm_combined_pwm	Uses FTM PWM functionality using two combined channels to light two LEDs on the board with opposite pulse width. The light's intensity is increased and decreased periodically.
	ftm_periodic_interrupt	Uses FTM Timer functionality to trigger an interrupt at a given period which toggles a LED.
	ftm_signal_measurement	Using one FTM instance the example application generates a PWM signal with variable frequency which is measured by another FTM instance configured in signal measurement mode.
	i2c_pal	Shows the usage of I2C PAL driver in both master and slave configurations using FLEXIO and LPI2C
	ic_pal	Uses IC Signal measurement to measure the frequency of a signal generated by another FTM instance on the same MCU.
	lpi2c_master	Shows the usage of the LPI2C driver in Master configuration
	lpi2c_slave	Shows the usage of the LPI2C driver in Slave configuration
	lpit_periodic_interrupt	Shows how to initialize the LPIT to generate an interrupt every 1 s. It is the starting point for any application using LPIT.
	lpspi_dma_master	The application uses on board instance of LPSPI, one in master configuration to communicate data via the SPI bus using DMA.
	lpspi_dma_slave	The application uses on board instance of LPSPI, one in slave configuration to communicate data via the SPI bus using DMA.
	lpspi_transfer_master	The application uses on board instance of LPSPI, one in master configuration to communicate data via the SPI bus
	lpspi_transfer_slave	The application uses on board instance of LPSPI, one in slave configuration to communicate data via the SPI bus
	lptmr_periodic_interrupt	Exemplifies to the user how to initialize the LPTIMER so that it will generate an interrupt every 1 second. To make the interrupt visible a LED is toggled every time it occurs.
	lptmr_pulse_counter	Shows the LPTIMER pulse count functionality by generating an interrupt every 4 rising edges.
	lpuart_echo	Simple example of a basic echo using LPUART.

	mpu_memory_protection	Configures MPU to protect a memory area and demonstrates that read access is correctly restricted.
	mpu_pal_memory_protection	The purpose of this demo application is to show you how to configure and use the Memory Protection Unit PAL
	oc_pal	Shows the Periodic Event Generation functionality of the OC_PAL
	pdb_periodic_interrupt	Configures the Programmable Delay Block to generate an interrupt every 1 second. This example shows the user how to configure the PDB timer for interrupt generation. The PDB is configured to trigger ADC conversions in ADC_HwTrigger_Example.
	power_mode_switch	Demonstrates the usage of Power Manager by allowing the user to switch to all power modes available.
	pwm_pal	Shows the PWM PAL functionality using FTM module found on the S32K116 using the S32 SDK API.
	security_pal	This is an application created to show the generation of rnd and CBC encryption/decryption of a string.
	rtc_alarm	Show the frequently used RTC use cases such as the generation of an interrupt every second and triggering an alarm.
	spi_pal	The purpose of this application is to show you how to use the LPSPI and FLEXIO Interfaces on the S32K116 using the S32 SDK API. The application uses one board instance of LPSPI in slave configuration and one board instance of FLEXIO in master configuration to communicate data via the SPI bus using interrupts.
	timing_pal	The purpose of this application is to show you how to use the TIMING PAL over LPIT, LPTMR and FTM timers on the S32K116 using the S32 SDK API. The application uses one board instance of LPIT, LPTMR and FTM to periodically toggle 3 leds.
	trgmux_lpit	The purpose of this demo application is to show you how to use the Trigger MUX Control of the S32K116 MCU with this SDK.
	uart_pal_echo	The purpose of this demo is to show the user how UART PAL works over FLEXIO_UART or LPUART peripherals. The user can choose whether to use FLEXIO_UART or LPUART. The board sends a welcome message to the console with further instructions.
	wdog_interrupt	Shows the basic usage scenario and configuration for the Watchdog.
	wdg_pal_interrupt	The purpose of this driver application is to show the user how to use the WDG PAL from the S32K116 using the S32 SDK API. The examples uses the SysTick timer from the ARM core to refresh the WDG PAL counter for 30 times. After this the WDG PAL counter will expire and the CPU will be reset.

Demos	hello_world	This is a simple application created to show the basic configuration with S32DS
	hello_world_mkf	This is a simple application created to show the basic configuration with makefile for the supported compilers
	flexcan_encrypted	Uses two boards to demonstrate FlexCAN functionality with Flexible Data Rate on. LEDs on a board are toggled depending on the buttons actioned on the other board. Also demonstrates the use of SBC driver to configure the CAN transceiver from EVB board. The application is configured to use CSEc to encrypt the data on security enabled parts.
	lin_master	This demo application shows the usage of LIN stack in master mode.
	lin_slave	This demo application shows the usage of LIN stack in slave mode.

5. Supported hardware and compatible software

5.1 CPUs

- PS32K116GMLF 0N96V 32 pin variant
- PS32K116GMLF 0N96V 48 pin variant

The following processor reference manual has been used to add support:

- S32K1XXRM Rev. 6, 12/2017

5.2 Boards

- S32K-MB with mini module S32K1XXCVD-Q48
- S32K116EVB-Q048 REV X1

5.3 Compiler and IDE versions

- GreenHills compiler v. 2015.1.4
- IAR compiler v. 7.50.3
- GCC compiler for ARM v. 4.9.3 20150529
- S32 Design Studio v2018.R1 IDE

5.4 Debug Probes

- SEGGER J-Link (with SEGGERGDB Server)
- P&E Multilink (with P&E GDB Server)

6. Known issues and limitations

6.1 Installer

- Due to an installer issue, before installing the new SDK, please make sure that the S32SDK_PATH environment variable is empty. This can be done either manually or by uninstalling the previous SDK version using the uninstall provided.
- The uninstaller does not delete configuration files copied in S32 DS build.
- Custom install type is not fully supported, keep “AllPackages” selection in Choose Components page.

6.2 Drivers

CAN_PAL

- Rx FIFO filtering for format B and C does not work
- Cannot respond to remote frames
- Cannot abort transfers

CPU

- If main function is exited the CPU will remain blocked in an infinite loop from startup_S32K116.S
- Doxygen Documentation link is missing from the CPU ProcessorExpert Component.

EIM

- If more than 2 bits are flipped in DATA_MASK or CHECKBIT_MASK bitfields in EIM control registers, there is no guarantee in design what type of error is generated

FLASH

- If the application uses flash and CSEc drivers from different threads, it must ensure that they don’t execute commands at the same time. The flash and CSEc drivers don’t currently implement an inter-synchronization mechanism, so the application will have to ensure it if needed.

FLEXCAN

- FLEXCAN_DRV_AbortTransfer does not abort the transfer for the selected message buffer
- Driver does not handle Rx FIFO warning and overflow events properly; frames may be lost due to FIFO overflow.
- Rx FIFO filtering for format B and C does not work
- Cannot respond to remote frames

FlexIO_I2C

- No STOP condition is generated when aborting a transfer due to NACK reception.
- No clock stretching when the application does not provide data fast enough, so Tx underflows and Rx overflows are possible.
- There is a maximum limit of 13 bytes on the size of any transfer.
- The driver does not support multi-master mode. It does not detect arbitration loss condition.
- Due to device limitations, it is not always possible to tell the difference between NACK reception and receiver overflow.

Note: FLEXIO I2C issues described above are caused by Hardware limitations.

FlexIO_SPI

- The driver does not support back-to-back transmission mode for CPHA = 1

FTM

- Module can be used only in one mode. For example, this configuration is not possible: 4 channels of FTM0 run in PWM and 4 channel of FTM0 run in input capture.

LIN/LIN_STACK

- In transmitted or received data state, LIN node will count break field sequence as a framing error. So, when it receives a valid break and sync field in received or transmitted data, LIN slave node will not enter in the PID sub-state.3
- Received data is invalid when using LIN_DRV_ReceiveFrameDataBlocking function

LPI2C

- LPI2C_DRV_MasterAbortTransferData function can't abort a master receive transfer because the module sees the whole receive as a single operation and will not stop it even if the FIFO is reset.

LPIT

- LPIT component raises errors when the configured LPIT clock frequency is a non-integer number.

PINS

- PinSettings component will not issue warnings when pins routed by default are overwritten. This may impact the debug functionality when JTAG or SWD pins are silently rerouted when other functionality is selected for the shared pins.

7.4 Examples

- Running the FLASH driver example from the flash will secure the device. To unsecure the MCU a mass erase of the flash needs to be done.
- Redundant code for configuring pins can be found in the examples.

7. Compiler options

The example projects are using the first level of optimizations (low optimizations).

For exceptions from the following compiler settings, additional information can be found in the SDK documentation, Build Tools section.

7.1 IAR Compiler/Linker/Assembler Options

Table 7.1 IAR Compiler Options

Option	Description
-Ol	Low optimizations
-e	Allow IAR extensions
--cpu Cortex-M0+	Selects target processor: Arm Cortex M0+
--thumb	Selects generating code that executes in Thumb state.
--debug	Include debug information
-D<cpu_define>	Define a preprocessor symbol for MCU
-warnings_are_errors	Treat code warnings as errors

Table 7.2 IAR Assembler Options

Option	Description
--cpu Cortex-M0+	Selects target processor: Arm Cortex M0+
--thumb	Selects generating code that executes in Thumb state.
-DSTART_FROM_FLASH	Mandatory when flash target is used

Table 7.3 IAR Linker Options

Option	Description
--cpu Cortex-M0+	Selects target processor: Arm Cortex M0+
--thumb	Selects generating code that executes in Thumb state.
--map <map_file>	Produce a linker memory map file
--entry Reset_Handler	Make the symbol Reset_Handler be treated as a root symbol and the start label of the application
--config <linker_file.icf>	Use the specified linker file

7.2 GCC Compiler/Linker/Assembler Options

Table 7.4 GCC Compiler Options

Option	Description
-mcpu=cortex-m0plus	Selects target processor: Arm Cortex M0+
-mthumb	Selects generating code that executes in Thumb state.
-O1	Optimize
-funsigned-char	Let the type char be unsigned, like unsigned char
-funsigned-bitfields	Bit-fields are signed by default
-fshort-enums	Allocate to an enum type only as many bytes as it needs for the declared range of possible values.
-ffunction-sections	Place each function into its own section in the output file
-fdata-sections	Place data item into its own section in the output file
-fno-jump-tables	Do not use jump tables for switch statements
-std=c99	Use C99 standard
-g	Generate debug information
-D<cpu_define>	Define a preprocessor symbol for MCU
-mfloat-abi=soft	Use software FP
-Wall	Produce warnings about questionable constructs
-Wextra	Produce extra warnings that -Wall
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types.
-pedantic	Issue all the warnings demanded by strict ISO C
-Wunused	Produce warnings for unused variables
-Werror	Treat warnings as errors
-Wsign-compare	Produce warnings when comparing signed type with unsigned type

Table 7.5 GCC Assembler Options

Option	Description
-mcpu=cortex-m0plus	Selects target processor: Arm Cortex M0+
-mthumb	Selects generating code that executes in Thumb state.
-mfloat-abi=soft	Use software FP
-Wall	Produce warnings about questionable constructs
-Wextra	Produce extra warnings that -Wall
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types.
-pedantic	Issue all the warnings demanded by strict ISO C
-Werror	Treat warnings as errors
-x assembler-with-cpp	Preprocess assembly files
-DSTART_FROM_FLASH	Mandatory when flash target is used

Table 7.6 GCC Linker Options

Option	Description
-mcpu=cortex-m0plus	Selects target processor: Arm Cortex M0+
-mthumb	Selects generating code that executes in Thumb state.
--entry=Reset_Handler	Make the symbol Reset_Handler be treated as a root symbol and the start label of the application
-T<linker_file.ld>	Use the specified linker file
-mfloat-abi=hard	Use FPU instructions
-mfpu=fpv4-sp-d16	Specify the FPU variant
-Xlinker –gc-sections	Remove unused sections
-Wl, -Map=<map_file>	Produce a map file
-lgcc	Link libgcc
-lc	Link C library
-lm	Link Math library

7.3 GHS Compiler/Linker/Assembler Options

Table 7.7 GHS Compiler Options

Option	Description
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+
-thumb	Selects generating code that executes in Thumb state.
-fsoft	Use software FP
-c99	Use C99 standard
--gnu_asm	Enables GNU extended asm syntax support
-Ogeneral	Optimize
-gdwarf-2	Generate DWARF 2.0 debug information
-G	Generate debug information
-D<cpu_define>	Define a preprocessor symbol for MCU
--quit_after_warnings	Treat warnings as errors
-Wimplicit-int	Produce warnings if functions are assumed to return int
-Wshadow	Produce warnings if variables are shadowed
-Wtrigraphs	Produce warnings if trigraphs are detected
-Wundef	Produce a warning if undefined identifiers are used in #if preprocessor statements

Table 7.8 GHS Assembler Options

Option	Description
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+
-fsoft	Use software FP
-preprocess_assembly_files	Preprocess assembly files
DSTART_FROM_FLASH	Mandatory when flash target is used

Table 7.9 GHS Linker Options

Option	Description
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+
-thumb	Selects generating code that executes in Thumb state.
-entry=Reset_Handler	Make the symbol Reset_Handler be treated as a root symbol and the start label of the application
-T<linker_file.ld>	Use the specified linker file
-map=<map_file>	Produce a map file
-larch	Link architecture specific library

Note: The symbol <linker_file> must be replaced with the corresponding path and linker file name per device, memory model and target compiler.

E.g. *C:\NXP\S32_SDK\platform\devices\S32K116\linker\gcc\S32K116_17_flash.ld* - for S32K116, 17 KB of SRAM and Flash target on GCC.

Symbol <map_file> shall be replaced with the desired map file name.

Symbol <cpu_define> shall be replaced with CPU_S32K116 for S32K116.

8. Acronyms

Acronym	Description
EAR	Early Access Release
JRE	Java Runtime Environment
EVB	Evaluation board
PAL	Peripheral Abstraction Layer
RTOS	Real Time Operating System
PEx	Processor Expert Configurator
PD	Peripheral Driver
RTM	Ready to Manufacture
S32DS	S32 Design Studio IDE
SDK	Software Development Kit
SOC	System-on-Chip
AMMCLib	Automotive Math and Motor Control Library
SCST	Structural Core Self Test

9. Version Tracking

Date (dd-Mmm-YYYY)	Version	Comments	Author
2-Feb-2018	1.0	First version for EAR 1.8.7	Rares Vasile